

# vCamera Final Report

Yahya Hassanzadeh

May 2016

<https://www.youtube.com/watch?v=9Ximl20pe2s>

## 1 Problem Definition

The smartphones' cameras are not always very easy to use. Their interfaces in some situations act as a burden for the user. This causes the user to lose some portion of his control over the interface.

Smartphones' cameras have dozens of adjustable parameters, like the flash-light, effect, white balance, scene, etc. They usually provide a control interface for the user to have an excellent access control on the camera parameters. Putting these too many adjustable parameters into menus and sub-menus makes the user not to have an easy time for surfing among them.

There is a common case among almost all smart phone cameras which makes the user traverse almost a long path consist of at least three menus and sub-menus for adjusting a single parameter of the camera. This procedure is both times consuming and hard to remember. Although users always prefer to use a particular limited set of adjustments, they need to follow the paths between menus and sub-menus each time they want to adjust a single parameter.

The situation is getting worse when the user switches from one smartphone model to another one. The camera apps are specialized by the manufacturer of the phone. Manufacturers usually prefer to launch a different interface for each model. Each brand new phone comes with almost a unique camera interface or menu organization. As an example, Figure 1 and 2 show the camera application's interface of LG G3 and LG G4, respectively. As it is shown in these figures, even though these two phones are from the same generation, their camera interfaces have a noticeable difference with each other. This difference is getting more evident when we compare two cameras from two different brands. For instance, Figure 3 shows the camera interface of the Samsung Galaxy S5, which apparently differs from the LG brands in the sense of menu organization.

A simple solution to ease up the user's experience with the camera interface is to build a single camera interface and distribute it to the market of smartphones. In this way, besides the smartphones' embedded camera, the user can benefit from a single unique application that its menu structure is consistent



Figure 1: The LG G3 Camera Interface



Figure 2: The LG G4 Camera Interface

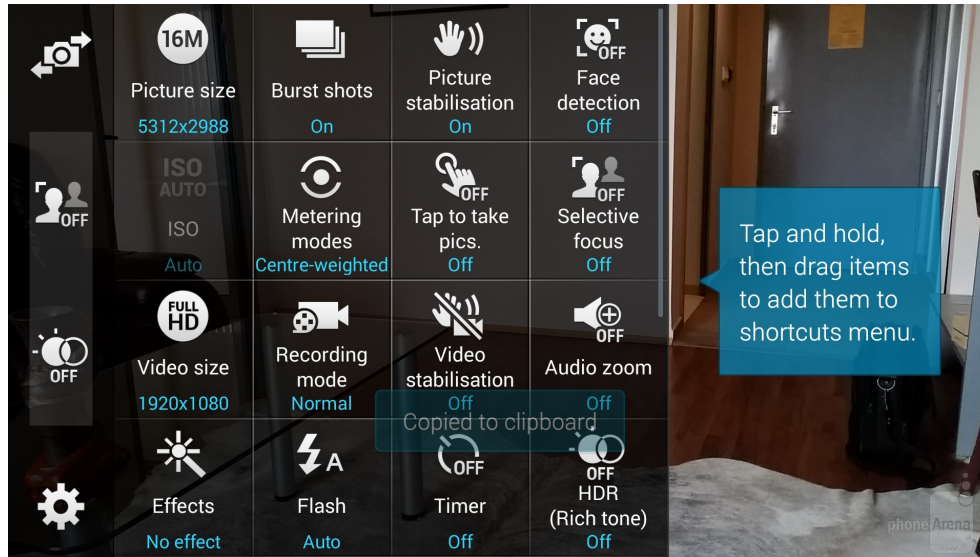


Figure 3: The Samsung S5 Camera Interface

with the phones and brands. Once the user gets familiar with this single camera application, it never faces the annoying experience of wondering among the apps.

This shared app over the various brands eases the experience of the user with the smartphones' camera. As the user never faces an unfamiliar camera interface. However, she still needs to traverse the paths between the menus and sub-menus to set a single parameter of the camera.

Likewise, there are situations where the user cannot interact with the camera interface correctly. On a sunny beach when the phone screen is more specular, finding the proper menus and submenus or even capturing a single photo is not very easy.

Taking a selfie photo is sometimes hard. As it needs the user to correctly pose the camera and press the shutter simultaneously with one hand which results in degradation of user's control level over the camera. This kind of adjustments sometimes causes jaggy photos.

Users commonly define a timer when there is no photographer or volunteer in the scene. Hence, they fire the timer, run to the scene and pose. Normally they have a maximum time of 10 seconds to get back to the scene and pose, which is sometimes too short. Also, in taking group photos consistent posing of all the group at the same time was difficult to manage. After each shooting was done, to take another picture, the user should come back to the camera and reset the timer.

To overcome all the problem mentioned in section 1, a voice-controlled camera has been proposed for the Android platform. In the project proposal, the

app was named as VCC, which stands for Voice Controlled Camera. But Later on, in the implementation phase, the name was changed to vCamera.

## 2 Novelties

By interacting with user over the voice, vCamera can offer the following services:

- User can adjust the common features of an embedded camera using voice. The set of features is effect, flashlight, focus mode, scene mode and white balance mode. **(Novel Feature)**
- User can take a photo or captures a video by saying a predefined sentence: OK camera, take a photo, or OK camera take a video
- User can switch between the front and back camera by saying a predefined sentence: OK camera, switch! **(Novel Feature)**
- User can zoom in/out before taking a photo by saying a predefined sentence like OK camera zoom in/ out. **(Novel Feature)**
- User can take multiple photos by a voice command. For example: "OK Camera, take three photos". **(Novel Feature)**

## 3 Technical Aspects

Figure 4 portraits the pipeline of *vCamera*. The speech recognizer API captures the user's voice command. Speech recognizer API converts the voice command to a text string and sends that to the interpreter. The interpreter then finds the appropriate command and interacts with the camera module to perform that command. In the following sub-sections, the modules that constitute the *vCamera* are described in more details:

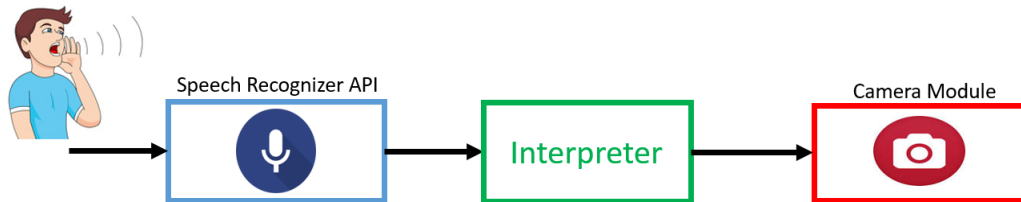


Figure 4: The *vCamera* pipeline

### 3.1 Speech Recognizer API

Based on the project proposal, the Google Voice API [1, 2] was supposed to be employed in the *vCamera*. It was implemented as a module and embedded into the *vCamera*. The following deficiencies were observed:

- The API does not support grammar. This causes the voice commands to be detected partially or wrongly.
- The user needed to trigger the voice recognition manually, by for example a pressing a button, or via an intent from the OS. Activating the API via a button degrades the *vCamera* functionality. Likewise, using an intent to do such operation makes the reaction speed of the application too slow.

To benefit from an always listening, grammar-based speech recognizer, the CMU Pocketsphinx [3] was employed. CMUSphinx benefits from a statistical hidden Markov model (HMM) [4] which describes the language. It offers many trained models for various acoustic conditions and different performance requirements.

As the model, we applied the *en-us-ptm* which represents an American English language. Also, we supplied the recognizer by the American English Dictionary *cmudict-en-us.dict*.

To recognize the users command a grammar was prepared and programmed to the recognizer module. Figure 5 shows the *vCamera*'s grammar for the voice commands.

As it is shown in this figure, every command starts with the special phrase, *OK Camera*, and continues with a body, which is called the *rest*. The body can be either a one part command, a two parts command which starts with the word *set*, a query command which starts with the word *take*, a help command which starts with the word *help*, or a record command.

#### 3.1.1 onePart commands

This commands are the ones that have a single fixed part for example zoom in, zoom out, switch, etc. A complete command including onePart commands is like: "OK camera, zoom in".

#### 3.1.2 twoParts commands

The twoParts commands are coming after a *set* word and an arbitrary *the* word. They are basically for setting a parameter of the camera to a certain state. Therefore, a twoParts command has an object and a state and sets the object to the certain state. For the user to interact more naturally, an arbitrary word *to* is also acceptable between the object and state. A complete command including twoParts commands is like: "OK camera, set the flashlight to auto".

```

#JSGF V1.0;

grammar menu;

public <basicCommand> = <start> <rest>;
<rest> = <onePart> | set (the)* <twoPart> | take <queryPart> | help <object> | record (a)* video;
<start> = ok camera;
<onePart> = zoom in | zoom out | switch | open gallery;
<twoPart> = <object> (to)* <state>;
<object> = effect mode | flash light | focus mode | scene mode | white balance mode | setting | command;
<queryPart> = <number> photos | (a)* photo;
<number> = one | two | three | four | five | six | seven | eight | nine | ten;
<state> = off |
        on |
        auto |
        continuous picture |
        continuous video |
        depth |
        fixed |
        infinity |
        close |
        black board |
        mono |
        white board |
        negative |
        none |
        solar |
        action |
        candle |
        firework |
        beach |
        landscape |
        portrait |
        snow |
        sport |
        balance |
        cloudy |
        daylight;

```

Figure 5: Voice Commands Grammar of *vCamera*

### 3.1.3 query command

A query command represents the operation of taking multiple photos, up to *ten*. A complete command including query commands is like: "OK camera, take a photo" or "OK camera, take ten photos".

### 3.1.4 help command

A help command causes showing help to the user respect to any of the objects. For example "OK camera, help scene mode". Help are shown as a toast message. For the user, to have a hint on the help, a hint on help message is shown in whenever the application is opened up or the phone is rotated. Figure 6 shows the graphical user interface of the *vCamera* as well as the hint on the help message as a toast.

On input, the speech recognizer module does a grammar test on input. The speech recognizer matches the input with the grammar and forwards the match -in the case of any match exists- as a string to the interpreter module. For example, *vCamera* does not react to the "OK camera" voice command. Since there is no match in the grammar for it. It is right that every command should be started by the "OK camera" phrase, but the voice command should then be continued by a proper body message.

## 3.2 Interpreter Module

The interpreter module works between the camera and speech recognizer modules, it receives the matched command string from the speech recognizer module. Then, it tries to interpret it as a command by a simple switch-case statement. As the proper command was found, and the input command string satisfied the command arguments, the command is executed based on the desired arguments. For example, the phrase "OK camera set the flashlight to depth" passes the grammar match successfully. But it cannot match to a proper command. Since depth is not a proper argument for the flashlight.

## 3.3 Camera Module

Camera module is based on the android camera library [5]. It provides the following functionality:

- Initializing an instance of camera object.
- Connecting the camera object to one of the available cameras of the Android device.
- Reads and writes the camera parameters from/to the camera device, respectively.
- Reads the preview from the camera device for the further processes.
- Interacts with the camera device.

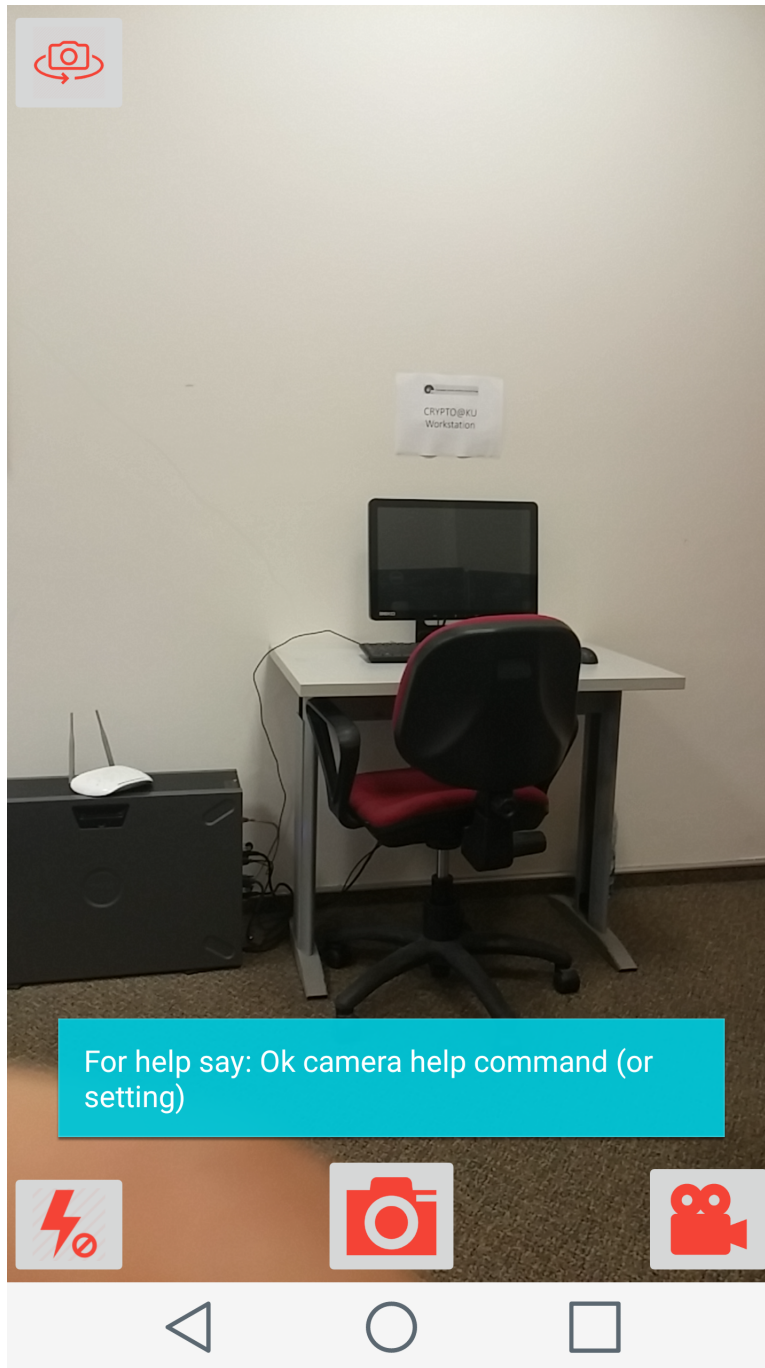


Figure 6: *vCamera*'s interface with the hint on help as a toast message

The camera module also embeds two smaller modules: Preview Module and Media Recorder Module.

### 3.3.1 Preview Module

Camera applications usually provide a preview screen for the user to have a view of the camera's viewing volume. Android camera class by itself does not provide such features. To provide a screen preview for the user, a preview module has been implemented as a separated class (*CameraPreview.java*). This class directly interacts with a Frame Layout [6]. The Frame Layout lays on the interface and continuously shows a live stream of the camera viewing volume.

Basically, the (*CameraPreview.java*) class inherits from the (*SurfaceView*) class [7] and implements a call back to it. A surface view object provides a drawing surface. The *SurfaceView* class takes care of placing the surface at the correct location on the screen.

Camera devices provide a preview based on the size of phone's screen. However, they do not preserve the aspect ratios between the objects in the scene. Also, regardless of phone's orientation, camera device always provides a portrait preview. To handle this issues, a function has been implemented to take care of the aspect ratio and orientation of preview screen based on the screen size and rotation degree of the phone, respectively.

In summary, the preview module does the followings:

- Provides a preview of camera's viewing volume for the user.
- Starts and stops capturing the preview.
- Feeds a preview to the media recorder for capturing a video.

### 3.3.2 Media Recorder Module

Recording a video has a different procedure from capturing a photo. To capture a video the following procedure should be done in order:

- Initializing a new media recorder object.
- Unlocking the camera.
- Binding the media recorder with the camera object.
- Setting the audio and video sources of the camera object.
- Setting the quality of the recorded video.
- Setting an output file for the camera object.
- Binding the media recorder object to the preview surface of the camera.

### 3.4 Differences From the Proposal

When the media recorder starts for capturing the video, it takes the mic under control and makes it inaccessible for others. Therefore, stopping the media recorder was not feasible by sound. A touch button has been designed for that. Except this issue, no other functionality was claimed in the project proposal that was not supported by the *vCamera*. However, as it is described in the Section 6, due to the majority of user’s suggestions, a few minor functionalities were implemented differently to fit the user’s desires.

## 4 Technology Used

*vCamera* has been implemented by Android Studio IDE for the Android platform. The minimum API requirement is 14, and the target API is 23. The latest release of the speech recognizer unit targeted the API 23 and was before the time that API higher than 23 was released. This makes the *vCamera* to be incompatible with APIs higher than 23 (e.g, Android 6.0). Also, the speech recognizer does not work properly in a very noisy environments.

## 5 Related Works

The voice-controlled camera Android applications that were distributed over the Google Play Store are categorized into two sub-groups: speech-based and sound-based applications.

### 5.1 Speech-based camera applications

Speech-based camera applications are defined as the ones that user can control at least one functionality of the camera via a voice command. Considering this definition, mainly two speech-based camera applications exist in the Google Play Store: Voice Camera [8] and Voice Power Camera [9]. All of these applications use a predefined command for capturing a photo or video, and that is the main functionality of them. In the other word, basically, except capturing a photo or recoding a video, almost none of the other operations of the camera can be controlled by voice.

These applications differ from each other in GUI, landscape/portrait photography support, supporting front/back/both camera(s) and the potency to change the default voice command sentence.

### 5.2 Sound-base camera applications

The general definition of the speech-based camera applications is also valid for the sound-based case. Except that in the sound-base applications, instead of a voice command as a predefined sentence, a sound command like a whistle sound is applied. Voice Selfie [10], Easy Selfie [11], Clap to take selfie [12] and

App Name	Category	Voice Control	Trigger	Timeout	Query Support
Voice Camera [8]	Speech-Based	No	Button	Yes	No
Voice Power Camera [9]	Speech-Based	No	Voice	No	No
Voice Selfie [10]	Sound-Based	No	Voice	No	No
Easy Selfie [11]	Sound-Based	No	Voice	No	No
Clap to take selfie [12]	Sound-Based	No	Voice	No	No
Whistle camera [13]	Sound-Based	No	Voice	No	No
<b>vCamera</b>	<b>Speech-Based</b>	<b>Yes</b>	<b>Voice</b>	<b>No</b>	<b>Yes</b>

Table 1: Comparison of related works with vCamera

Whistle camera [13] are the sound-based camera applications available in the Google Play Store. Similar to the voice base camera applications, in sound-based applications, the user can only take a photo or capture a video by generating a noise and does not have any other sound-based control over the camera.

Table 1 shows a comparison between various voice interaction cameras available in the Google Play Store as well as the vCamera. The voice control column, corresponds to the ability of controlling the camera parameters by voice except taking a photo or capturing a video. The trigger column corresponds to the activation method of the camera’s voice system that is whether by means of a voice or a button. The voice recognition system may put a time limitation for the user to speak with the camera. The timeout column represents this feature. Defining a query as the capability of asking the camera to take a number of repetitive photos, the query support column represents this feature of the voice controlled camera applications.

As a general observation, in the both speech-based and voice-based applications, the user can only capture a photo or video by a voice or sound command, respectively. And does almost not have any other voice/sound control over the camera. However, some of the applications like the Easy Selfie [11], offer a touch interface very similar to the common Android default camera apps. The user can then set some of the camera’s feature like the flash mode by means of touching the screen.

As the speech-based camera applications are preferred over the sound-base ones due to having a more natural and easy-going interaction with the user, and also are closer to the objectives of this project, they are considered as the state-of-the-arts for this project.

## 6 Evaluation

The evaluation of *vCamera* was done in two phase: initial and final phases. 20 users where selected and divided into two groups of 10. One group was employed in the initial evaluation and the other one in the final.

## 6.1 Initial Evaluation

In this phase, users were invited to interact with the *vCamera* application. Then they were asked to tell their suggestions about the *vCamera* and report the bugs. The reported bugs were fixed, and the suggestions were obtained the vote of majority where applied as the followings:

- In the initial phase, users should use the "Action" and "Capture" words to record a video or take a photo. Based on the user's suggestions, these two were changed to "Record a video" and "Take a photo", respectively.
- In the initial phase user could determine the time interval when they want to make a query. For example, they could say "OK Camera, take three photos in 3 seconds." The majority of users where agreed on eliminating the time intervals and convert it to a fixed time interval. Thus, the time interval argument was eliminated from the *vCamera*. Now it can support the queries like "OK Camera, take three photos".
- In the initial phase, there was a separate grammar for the flashlight. To turn the camera on or off, the proper command was "OK Camera, turn on/off flashlight". Speech recognizer sometimes confused this command with "OK Camera, take one photo". The majority of users suggested unifying this command with the *set* command.

## 6.2 Final Evaluation

After the initial evaluation was done, bugs were fixed, and proper changes were made, the final evaluation was performed. A two minutes video were captured and shared with a group of users accompanied with the application APK, and an on-line anonymous evaluation form. Users had around a week to interact with the application and submit their evaluation form anonymously. Followings are the list of questions were asked from the users:

1. How long have you been working with Android?
2. How frequently do you use your phone's camera?
3. Do you prefer to use a voice-controlled camera? (This is a yes/no question)
4. How often did you face bugs during the interaction?
5. How easy to understand the application?
6. How smooth was the workflow?
7. How was the voice command recognition?
8. How much did you satisfy with the voice command recognition?
9. Did you have an easy time with the application?

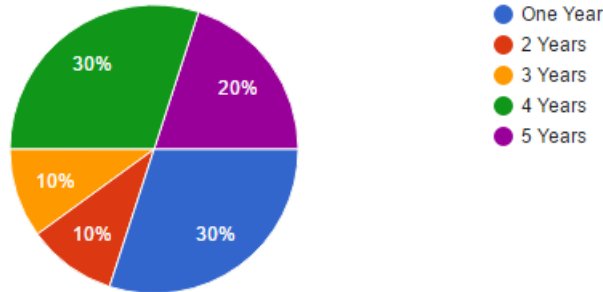


Figure 7: Android experience distribution of the final evaluation phase's users

10. Did you enjoy working with *vCamera*?
11. How much do you think about the usefulness of the application?
12. Did you feel any limit during the interaction?
13. How much did you satisfy of your experience with *vCamera*?

#### 6.2.1 Users' Android Experience

Figure 7 shows the Android experience distribution of the users in the final evaluation phase. On the average, users had 3 years of interacting with Android devices. Figure 8 shows the frequency distribution of camera usage among the users. About 80% of users preferred to have a voice controlled camera application.

#### 6.2.2 vCamera Bugs

Figure 9 depicts the correctness of application in the terms of facing bugs. Based on this figure, on the average, a user faced 3.3 bugs while interacting with the *vCamera*.

#### 6.2.3 Understanding Easiness

About 70% of the users declared that *vCamera* was easy to understand. The rest 30% stated that it is normal.

#### 6.2.4 Work Flow Easiness

One of the most important parts of the evaluation was to measure the easiness of the *vCamera* for the user. To catch the random answers, this question was asked in two different ways and two different places on the evaluation form: Questions number 6 and 12. Users responded to these questions with the same

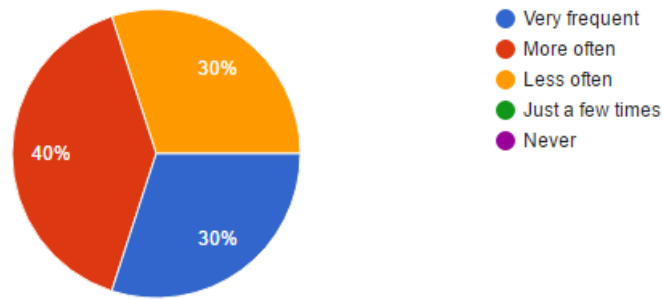


Figure 8: Camera usage frequency distribution of the final evaluation phase's users

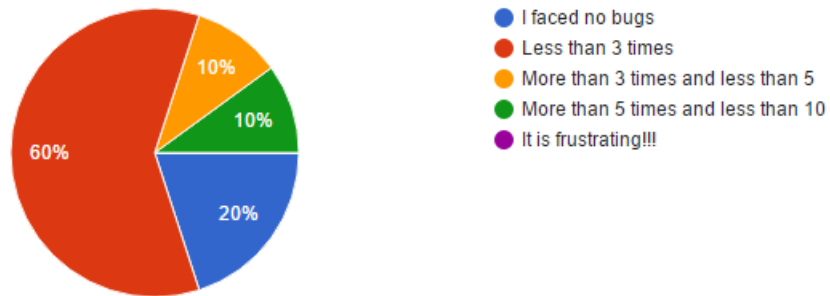


Figure 9: General error reports of the final evaluation phase's users

answers. Around 60% believed that *vCamera* is an easy going app. While the rest 40% believed that it is neither easy nor hard.

### 6.2.5 Users' Satisfaction

The users satisfactory from the *vCamera* was also asked in two different ways in questions 10 and 13, and users provided similar answers to both questions. Around 80% of users gave the satisfactory point to *vCamera*. 10% gave neutral grade and 10% gave the unsatisfactory grade. Also, around 80% of the users voted on the usefulness of the *vCamera*.

### 6.2.6 Speech Recognition Evaluation

Figure 10 shows the performance of the speech recognition unit from the view-point of users. This topic was also asked in three different ways as the questions number 7, 8 and 9. The responses to questions 7 and 8 are not the same in the first glimpse. However, putting a finer grain shows an acceptable similarity between the answers of this two questions. This similarity makes the answer to the question 9 more reliable. Based on this figure 3.3 voice commands of the users, on the average, were not recognized by the *vCamera*. This result is similar to the expected number of errors. After further investigation, it became clear that all of the users counted the number of fails in the speech recognition as an error as well.

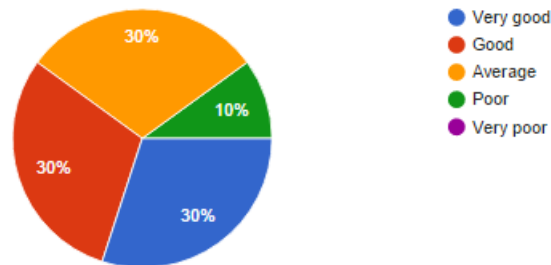
## 6.3 Additional Evaluation

In addition to the 13 questions that were asked from the users, another extra question that was not listed in the proposal was also asked. In that question, users were asked to compare the *vCamera* with the Voice Camera [8] from the Google Play Store. Voice camera is speech-based camera application which due to its fascinating GUI was supposed to be the best counterpart for *vCamera*. This was not a compulsory question to answer, so 50% of users did not answer to this question. However, the rest 50% who had installed this app on their devices still preferred the *vCamera*

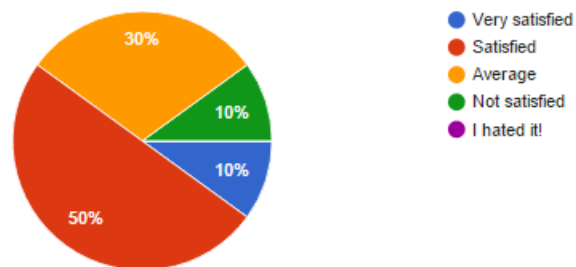
## References

- [1] Google. Voice action. [Online]. Available: <https://developers.google.com/voice-actions/>
- [2] ——. Overview of the voice interaction api. [Online]. Available: <https://developers.google.com/voice-actions/interaction/>
- [3] CMU. Cmu sphinx. [Online]. Available: <http://cmusphinx.sourceforge.net/wiki/tutorialandroid>

How was voice command recognition? (10 responses)



How much did you satisfied with the voice command recognition? (10 responses)



How many of your commands were not recognized by vCamera? (10 responses)

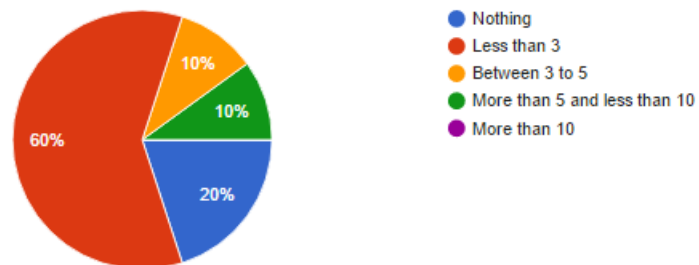


Figure 10: Performance of the speech recognition module

- [4] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.
- [5] Google. Android camera api guide. [Online]. Available: <https://developer.android.com/guide/topics/media/camera.html>
- [6] ——. Android frame layout api. [Online]. Available: <https://developer.android.com/reference/android/widget/FrameLayout.html>
- [7] ——. Android surface view. [Online]. Available: <https://developer.android.com/reference/android/view/SurfaceView.html>
- [8] “Voice camera: <https://play.google.com/store/apps/details?id=com.jayk.dev.noisecamera>.”
- [9] “Voice powered camera: <https://play.google.com/store/apps/details?id=com.morningshine.voicecamera>.”
- [10] “Voice selfie: <https://play.google.com/store/apps/details?id=com.amazinglocks.voiceselfie>.”
- [11] “Easy selfie: <https://play.google.com/store/apps/details?id=com.pojkarsoft.easyselfie>.”
- [12] “Clap to take selfies: <https://play.google.com/store/apps/details?id=com.mobmatrix.tool.selfiecamera>.”
- [13] “Whistle camera: <https://play.google.com/store/apps/details?id=com.dreambit.whistlecamera>.”